Movielens Case Study

Import libraries

```
[35] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
```

Import Dataset

Import Movies Dataset

```
[36] dfMovies = pd.read_csv("movies.dat",
                            sep="::",names=["MovieID","Title","Genres"],engine='python')
     dfMovies.head()
```

	MovieID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

Import Ratings Dataset

```
[37] dfRatings = pd.read_csv("ratings.dat",
                             sep="::",names=["UserID","MovieID","Rating","Timestamp"],engine='python')
     dfRatings.head()
```

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

Import Users Dataset

```
[38] dfUsers = pd.read_csv("users.dat",
                           sep="::",names=["UserID","Gender","Age","Occupation","Zip-code"],engine='python')
     dfUsers.head()
```

	UserID	Gender	Age	Occupation	Zip-code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

shape of the Dataset

```
[39] print(dfMovies.shape)
     print(dfRatings.shape)
     print(dfUsers.shape)
```

```
(3883, 3)
(1000209, 4)
(6040, 5)
```

Create a new dataset [Master_Data]

```
[40] dfMovieRatings = dfMovies.merge(dfRatings,on='MovieID',how='inner')
     dfMovieRatings.head(5)
```

	MovieID	Title	Genres	UserID	Rating	Timestamp
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268
1	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008
2	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496
3	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952
4	1	Toy Story (1995)	Animation|Children's|Comedy	10	5	978226474

```
[41] # to check whether merging does not changes any dataset
     dfMovieRatings.shape
```

```
(1000209, 6)
```

```
[42] dfMaster = dfMovieRatings.merge(dfUsers,on="UserID",how='inner')
     dfMaster.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067

```
[43] dfMaster.to_csv("Master.csv")
```

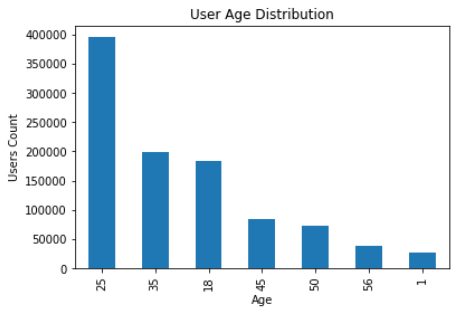
Explore the datasets using visual representations

User Age Distribution

```
[44] # Users with Different Age Groups
     dfMaster['Age'].value_counts()
```

```
25    395556
35    199003
18    183536
45     83633
50     72490
56     38780
1      27211
Name: Age, dtype: int64
```

```
[45] # Plot for users with different age groups
     dfMaster['Age'].value_counts().plot(kind='bar')
     plt.xlabel("Age")
     plt.title("User Age Distribution")
     plt.ylabel('Users Count')
     plt.show()
```



User rating of the movie “Toy Story”

```
[46] # Toy Story
     toystoryRating = dfMaster[dfMaster['Title'].str.contains('Toy Story') == True]
     toystoryRating
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067

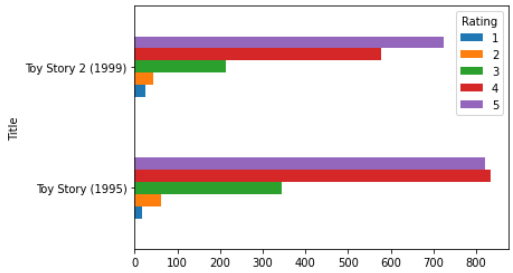
50	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	1	4	978302174	F	1	10	48067
53	1	Toy Story (1995)	Animation|Children's|Comedy	6	4	978237008	F	50	9	55117
124	1	Toy Story (1995)	Animation|Children's|Comedy	8	4	978233496	M	25	12	11413
263	1	Toy Story (1995)	Animation|Children's|Comedy	9	5	978225952	M	25	17	61614
...	...	...	...	...	...	...	...	...	...	...
998988	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	3023	4	970471948	F	25	7	92108
999027	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5800	5	958015250	M	35	18	90804
999486	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	2189	4	974607816	M	1	10	60148
999869	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	159	4	989966944	F	45	0	37922
1000192	3114	Toy Story 2 (1999)	Animation|Children's|Comedy	5727	5	958492554	M	25	4	92843

3662 rows × 10 columns

```
[47] toystoryRating.groupby(["Title","Rating"]).size()
```

```
Title               Rating
Toy Story (1995)    1          16
                    2          61
                    3         345
                    4         835
                    5         820
Toy Story 2 (1999)  1          25
                    2          44
                    3         214
                    4         578
                    5         724
dtype: int64
```

```
[48] toystoryRating.groupby(["Title","Rating"]).size().unstack().plot(kind='barh',stacked=False,legend=True)
     plt.show()
```

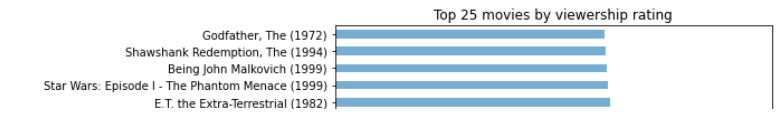


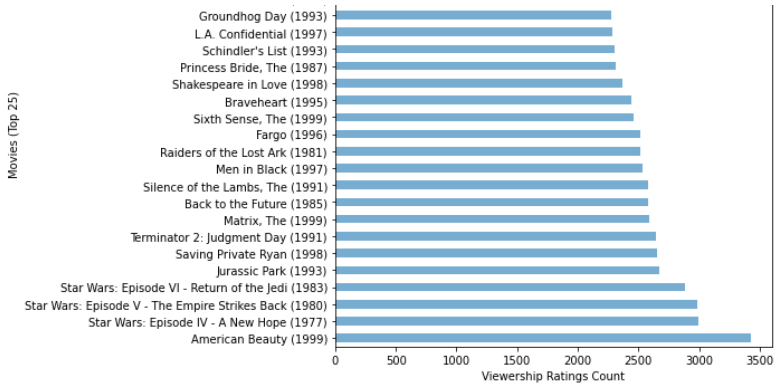
Top 25 movies by viewership rating

```
[49] dfTop25 = dfMaster.groupby('Title').size().sort_values(ascending=False)[:25]
     dfTop25
```

```
Title
American Beauty (1999)                              3428
Star Wars: Episode IV - A New Hope (1977)           2991
Star Wars: Episode V - The Empire Strikes Back (1980)    2990
Star Wars: Episode VI - Return of the Jedi (1983)   2883
Jurassic Park (1993)                                2672
Saving Private Ryan (1998)                          2653
Terminator 2: Judgment Day (1991)                   2649
Matrix, The (1999)                                  2590
Back to the Future (1985)                           2583
Silence of the Lambs, The (1991)                    2578
Men in Black (1997)                                 2538
Raiders of the Lost Ark (1981)                      2514
Fargo (1996)                                        2513
Sixth Sense, The (1999)                             2459
Braveheart (1995)                                   2443
Shakespeare in Love (1998)                          2369
Princess Bride, The (1987)                          2318
Schindler's List (1993)                             2304
L.A. Confidential (1997)                            2288
Groundhog Day (1993)                                2278
E.T. the Extra-Terrestrial (1982)                   2269
Star Wars: Episode I - The Phantom Menace (1999)    2250
Being John Malkovich (1999)                         2241
Shawshank Redemption, The (1994)                    2227
Godfather, The (1972)                               2223
dtype: int64
```

```
[50] dfTop25.plot(kind='barh',alpha=0.6,figsize=(7,7))
     plt.xlabel("Viewership Ratings Count")
     plt.ylabel("Movies (Top 25)")
     plt.title("Top 25 movies by viewership rating")
     plt.show()
```





Find the ratings for all the movies reviewed by for a particular user of user id = 2696

```
[51] userId = 2696
     userRatingById = dfMaster[dfMaster["UserID"] == userId]
     userRatingById
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code
991035	350	Client, The (1994)	Drama|Mystery|Thriller	2696	3	973308886	M	25	7	24210
991036	800	Lone Star (1996)	Drama|Mystery	2696	5	973308842	M	25	7	24210
991037	1092	Basic Instinct (1992)	Mystery|Thriller	2696	4	973308886	M	25	7	24210
991038	1097	E.T. the Extra-Terrestrial (1982)	Children's|Drama|Fantasy|Sci-Fi	2696	3	973308690	M	25	7	24210
991039	1258	Shining, The (1980)	Horror	2696	4	973308710	M	25	7	24210
991040	1270	Back to the Future (1985)	Comedy|Sci-Fi	2696	2	973308676	M	25	7	24210
991041	1589	Cop Land (1997)	Crime|Drama|Mystery	2696	3	973308865	M	25	7	24210
991042	1617	L.A. Confidential (1997)	Crime|Film-Noir|Mystery|Thriller	2696	4	973308842	M	25	7	24210
991043	1625	Game, The (1997)	Mystery|Thriller	2696	4	973308842	M	25	7	24210
991044	1644	I Know What You Did Last Summer (1997)	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210
991045	1645	Devil's Advocate, The (1997)	Crime|Horror|Mystery|Thriller	2696	4	973308904	M	25	7	24210
991046	1711	Midnight in the Garden of Good and Evil (1997)	Comedy|Crime|Drama|Mystery	2696	4	973308904	M	25	7	24210
991047	1783	Palmetto (1998)	Film-Noir|Mystery|Thriller	2696	4	973308865	M	25	7	24210
991048	1805	Wild Things (1998)	Crime|Drama|Mystery|Thriller	2696	4	973308886	M	25	7	24210
991049	1892	Perfect Murder, A (1998)	Mystery|Thriller	2696	4	973308904	M	25	7	24210
991050	2338	I Still Know What You Did Last Summer (1998)	Horror|Mystery|Thriller	2696	2	973308920	M	25	7	24210
991051	2389	Psycho (1998)	Crime|Horror|Thriller	2696	4	973308710	M	25	7	24210
991052	2713	Lake Placid (1999)	Horror|Thriller	2696	1	973308710	M	25	7	24210
991053	3176	Talented Mr. Ripley, The (1999)	Drama|Mystery|Thriller	2696	4	973308865	M	25	7	24210
991054	3386	JFK (1991)	Drama|Mystery	2696	1	973308842	M	25	7	24210

```
[52] #There is User in whole dataset who having particular user id = 2696
```

Feature Engineering

Find out all the unique genres

```
[53] #dfGenres = dfMaster[]
     dfGenres = dfMaster['Genres'].str.split("|")
     dfGenres
```

```
0                  [Animation, Children's, Comedy]
1         [Animation, Children's, Musical, Romance]
2                                         [Drama]
3              [Action, Adventure, Fantasy, Sci-Fi]
4                                   [Drama, War]
                         ...
1000204                        [Drama, Thriller]
1000205                [Comedy, Horror, Thriller]
1000206                        [Comedy, Romance]
1000207                       [Action, Thriller]
1000208                          [Action, Drama]
Name: Genres, Length: 1000209, dtype: object
```

```
[54] listGenres = set()
     for genre in dfGenres:
         listGenres = listGenres.union(set(genre))
```

```
[55] # All Unique genres
     listGenres
```

```
{'Action',
 'Adventure',
 'Animation',
 "Children's",
 'Comedy',
 'Crime',
 'Documentary',
 'Drama',
 'Fantasy',
 'Film-Noir',
 'Horror',
 'Musical',
 'Mystery',
 'Romance',
 'Sci-Fi',
 'Thriller',
 'War',
 'Western'}
```

Create a separate column for each genre category with a one-hot encoding (1 and 0) whether or not the movie belongs to that genre.

```
[56] ratingsOneHot = dfMaster['Genres'].str.get_dummies("|")
```

```
[57] ratingsOneHot.head()
```

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0

```
[58] dfMaster = pd.concat([dfMaster,ratingsOneHot],axis=1)
```

```
[59] dfMaster.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	Action	Adventure	Animation	Children's	Comedy
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067	0	0	1	1	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067	0	0	1	1	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067	0	0	0	0	0
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067	1	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067	0	0	0	0	0

```
[60] dfMaster.columns
```

```
Index(['MovieID', 'Title', 'Genres', 'UserID', 'Rating', 'Timestamp', 'Gender',
       'Age', 'Occupation', 'Zip-code', 'Action', 'Adventure', 'Animation',
       'Children's', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',
       'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi',
       'Thriller', 'War', 'Western'],
      dtype='object')
```

```
[61] dfMaster.to_csv("Final_Master.csv")
```

Determine the features affecting the ratings of any particular movie

```
[62] dfMaster[["title","Year"]] = dfMaster.Title.str.extract("(.)\s\((.\d+)",expand=True)
```

```
[63] dfMaster = dfMaster.drop(columns=["title"])
     dfMaster.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	Action	Adventure	Animation	Children's	Comedy
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067	0	0	1	1	1

1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067	0	0	1	1	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067	0	0	0	0	0
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067	1	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067	0	0	0	0	0

[64]
```
dfMaster.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000209 entries, 0 to 1000208
Data columns (total 29 columns):
 #   Column       Non-Null Count    Dtype 
---  ------       --------------    ----- 
 0   MovieID      1000209 non-null  int64 
 1   Title        1000209 non-null  object
 2   Genres       1000209 non-null  object
 3   UserID       1000209 non-null  int64 
 4   Rating       1000209 non-null  int64 
 5   Timestamp    1000209 non-null  int64 
 6   Gender       1000209 non-null  object
 7   Age          1000209 non-null  int64 
 8   Occupation   1000209 non-null  int64 
 9   Zip-code     1000209 non-null  object
 10  Action       1000209 non-null  int64 
 11  Adventure    1000209 non-null  int64 
 12  Animation    1000209 non-null  int64 
 13  Children's   1000209 non-null  int64 
 14  Comedy       1000209 non-null  int64 
 15  Crime        1000209 non-null  int64 
 16  Documentary  1000209 non-null  int64 
 17  Drama        1000209 non-null  int64 
 18  Fantasy      1000209 non-null  int64 
 19  Film-Noir    1000209 non-null  int64 
 20  Horror       1000209 non-null  int64 
 21  Musical      1000209 non-null  int64 
 22  Mystery      1000209 non-null  int64 
 23  Romance      1000209 non-null  int64 
 24  Sci-Fi       1000209 non-null  int64 
 25  Thriller     1000209 non-null  int64 
 26  War          1000209 non-null  int64 
 27  Western      1000209 non-null  int64 
 28  Year         1000209 non-null  object
dtypes: int64(24), object(5)
memory usage: 228.9+ MB
```

[65]
```
dfMaster['Year'] = dfMaster.Year.astype(int)
```

[66]
```
dfMaster['Movie_Age'] = 2000 - dfMaster.Year
dfMaster.head()
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	Action	Adventure	Animation	Children's	Comedy
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	F	1	10	48067	0	0	1	1	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	F	1	10	48067	0	0	1	1	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	F	1	10	48067	0	0	0	0	0
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	F	1	10	48067	1	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	F	1	10	48067	0	0	0	0	0

[67]
```
dfMaster['Gender'] = dfMaster.Gender.str.replace('F','1')
dfMaster['Gender'] = dfMaster.Gender.str.replace('M','0')
dfMaster['Gender'] = dfMaster.Gender.astype(int)
dfMaster.head()
```

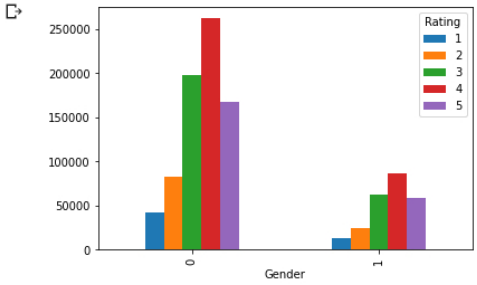
	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	Action	Adventure	Animation	Children's	Comedy
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	1	1	10	48067	0	0	1	1	1
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	1	1	10	48067	0	0	1	1	0
2	150	Apollo 13 (1995)	Drama	1	5	978301777	1	1	10	48067	0	0	0	0	0
		Star Wars: Episode IV													

3	260	Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	1	1	10	48067	1	1	0	0	0
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	1	1	10	48067	0	0	0	0	0

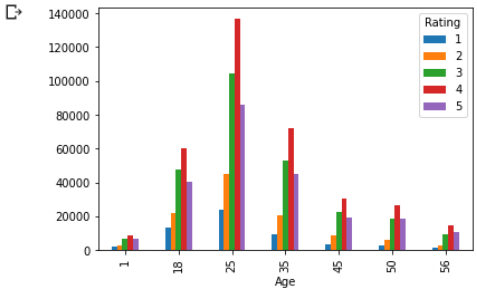
```
[113] dfGenderAffecting = dfMaster.groupby('Gender').size().sort_values(ascending=False)[:25]
      dfGenderAffecting
```

```
Gender
0    753769
1    246440
dtype: int64
```

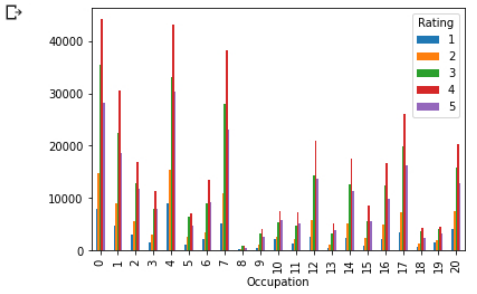
```
[69] dfMaster.groupby(["Gender","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
     plt.show()
```



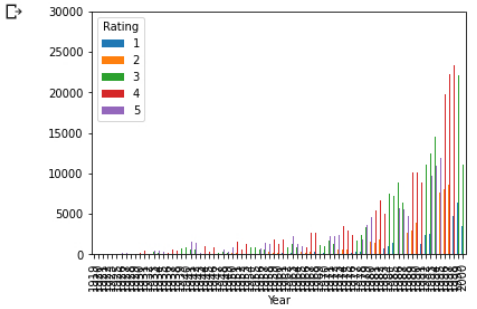

```
[70] dfMaster.groupby(["Age","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
     plt.show()
```



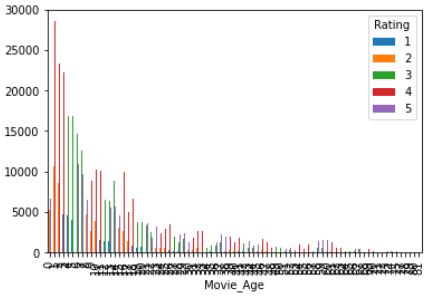

```
[71] dfMaster.groupby(["Occupation","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
     plt.show()
```




```
[72] dfMaster.groupby(["Year","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
     plt.show()
```




```
[73] dfMaster.groupby(["Movie_Age","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
     plt.show()
```



Develop an appropriate model to predict the movie ratings

```
[74] #First 500 extracted records
     first_500 = dfMaster[:1000]
     first_500
```

	MovieID	Title	Genres	UserID	Rating	Timestamp	Gender	Age	Occupation	Zip-code	Action	Adventure	Animation	Children's	Come
0	1	Toy Story (1995)	Animation|Children's|Comedy	1	5	978824268	1	1	10	48067	0	0	1	1	
1	48	Pocahontas (1995)	Animation|Children's|Musical|Romance	1	5	978824351	1	1	10	48067	0	0	1	1	
2	150	Apollo 13 (1995)	Drama	1	5	978301777	1	1	10	48067	0	0	0	0	
3	260	Star Wars: Episode IV - A New Hope (1977)	Action|Adventure|Fantasy|Sci-Fi	1	4	978300760	1	1	10	48067	1	1	0	0	
4	527	Schindler's List (1993)	Drama|War	1	5	978824195	1	1	10	48067	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	2384	Babe: Pig in the City (1998)	Children's|Comedy	18	2	978155233	1	18	3	95825	0	0	0	1	
996	2391	Simple Plan, A (1998)	Crime|Thriller	18	1	978155685	1	18	3	95825	0	0	0	0	
997	2394	Prince of Egypt, The (1998)	Animation|Musical	18	4	978154907	1	18	3	95825	0	0	1	0	
998	2402	Rambo: First Blood Part II (1985)	Action|War	18	2	978153894	1	18	3	95825	1	0	0	0	
999	2404	Rambo III (1988)	Action|War	18	2	978153977	1	18	3	95825	1	0	0	0	

1000 rows × 30 columns

```
[78] #Use the following features:movie id,age,occupation
     features = first_500[['MovieID','Age','Occupation']].values
     #Use rating as label
     labels = first_500[['Rating']].values
```

```
[80] features
```

```
array([[   1,    1,   10],
       [  48,    1,   10],
       [ 150,    1,   10],
       ...,
       [2394,   18,    3],
       [2402,   18,    3],
       [2404,   18,    3]], dtype=int64)
```

```
[81] labels
```

Importing Libraries Form ML

```
[107] from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.svm import SVC, LinearSVC
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.naive_bayes import GaussianNB
      from sklearn.linear_model import Perceptron
      from sklearn.linear_model import SGDClassifier
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

spliting dataset for model

```
[84] #Create train and test data set
train, test, train_labels, test_labels = train_test_split(features,labels,test_size=0.33,random_state=42)
```

Logistic Regression

```
[87] #Logistic Regression

logreg = LogisticRegression()
logreg.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
[88] Y_pred = logreg.predict(test)
acc_log = round(logreg.score(train, train_labels) * 100, 2)
acc_log
```

```
36.72
```

```
[ ]
```

Support Vector Machines

```
[90] #Support Vector Machines

svc = SVC()
svc.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
[91] Y_pred = svc.predict(test)
acc_svc = round(svc.score(train, train_labels) * 100, 2)
acc_svc
```

```
38.81
```

KNeighborsClassifier

```
[93] #K Nearest Neighbors Classifier

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the
  after removing the cwd from sys.path.
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

```
[94] Y_pred = knn.predict(test)
acc_knn = round(knn.score(train, train_labels) * 100, 2)
acc_knn
```

```
59.7
```

Naive bayes

```
[97] # Gaussian Naive Bayes
```

```
gaussian = GaussianNB()
gaussian.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/naive_bayes.py:206: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change
  y = column_or_1d(y, warn=True)
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
[98] Y_pred = gaussian.predict(test)
acc_gaussian = round(gaussian.score(train, train_labels) * 100, 2)
acc_gaussian
```

```
39.55
```

Perceptron

```
[100] perceptron = Perceptron()
perceptron.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
Perceptron(alpha=0.0001, class_weight=None, early_stopping=False, eta0=1.0,
           fit_intercept=True, max_iter=1000, n_iter_no_change=5, n_jobs=None,
           penalty=None, random_state=0, shuffle=True, tol=0.001,
           validation_fraction=0.1, verbose=0, warm_start=False)
```

```
[101] Y_pred = perceptron.predict(test)
acc_perceptron = round(perceptron.score(train, train_labels) * 100, 2)
acc_perceptron
```

```
34.33
```

Linear SVC

```
[102] # Linear SVC

linear_svc = LinearSVC()
linear_svc.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
          verbose=0)
```

```
[103] Y_pred = linear_svc.predict(test)
acc_linear_svc = round(linear_svc.score(train, train_labels) * 100, 2)
acc_linear_svc
```

```
37.76
```

Stochastic Gradient Descent

```
[105] # Stochastic Gradient Descent

sgd = SGDClassifier()
sgd.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:760: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please ch
  y = column_or_1d(y, warn=True)
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
```

```
[106] Y_pred = sgd.predict(test)
acc_sgd = round(sgd.score(train, train_labels) * 100, 2)
acc_sgd
```

```
37.61
```

Decision Tree

```
[108] # Decision Tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(train, train_labels)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
[109] Y_pred = decision_tree.predict(test)
      acc_decision_tree = round(decision_tree.score(train, train_labels) * 100, 2)
      acc_decision_tree
```

```
100.0
```

Random Forest

```
[110] # Random Forest

      random_forest = RandomForestClassifier(n_estimators=100)
      random_forest.fit(train, train_labels)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the
  after removing the cwd from sys.path.
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
[111] Y_pred = random_forest.predict(test)
      random_forest.score(train, train_labels)
      acc_random_forest = round(random_forest.score(train, train_labels) * 100, 2)
      acc_random_forest
```

```
99.85
```

Score of each model

```
[116] models = pd.DataFrame({
          'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
                    'Random Forest', 'Naive Bayes', 'Perceptron',
                    'Stochastic Gradient Decent', 'Linear SVC',
                    'Decision Tree'],
          'Score': [acc_svc, acc_knn, acc_log,
                    acc_random_forest, acc_gaussian, acc_perceptron,
                    acc_sgd, acc_linear_svc, acc_decision_tree]})
```

```
[117] models.sort_values(by='Score', ascending=False)
```

	Model	Score
8	Decision Tree	100.00
3	Random Forest	99.85
1	KNN	59.70
4	Naive Bayes	39.55
0	Support Vector Machines	38.81
7	Linear SVC	37.76
6	Stochastic Gradient Decent	37.61
2	Logistic Regression	36.72
5	Perceptron	34.33